Exact Incremental and Decremental Learning for LS-SVM

Wei-Han Lee¹, Bong Jun Ko¹, Shiqiang Wang¹, Changchang Liu¹, Kin K. Leung²

¹ IBM T. J. Watson Research Center, Yorktown Heights, NY, USA
² Imperial College London, UK
A day in the office of a machine learning (ML) engineer
A day in the office of a machine learning (ML) engineer

Here's the dataset you've been waiting for.

The Manager

The ML Engineer
A day in the office of a machine learning (ML) engineer

Here’s the dataset you’ve been waiting for.

Great! Now I can build a model.

The Manager

The ML Engineer
A day in the office of a machine learning (ML) engineer

The Manager

Here’s the dataset you’ve been waiting for.

The ML Engineer

Done!
The next day

The ML Engineer
The next day

Here’s another dataset we just acquired. We need this reflected in the model too.

The Manager

The ML Engineer
The next day

Here’s another dataset we just acquired. We need this reflected in the model too.

Ok, no problem. I’ll just need to add it to the database and run the training job again, so the model doesn’t forget the old dataset.

The Manager

The ML Engineer
The next day

Here’s another dataset we just acquired. We need this reflected in the model too.

Done! It took a little longer, but not a big deal...

The Manager

The ML Engineer
And so it goes on…

Here’s **yet another** dataset we just acquired. We need this reflected in the model too.
And so it goes on…

Here’s **yet another** dataset we just acquired. We need this reflected in the model too.

Aarggggh! Not again…

The Manager

The ML Engineer
And yet, even worse...

Oh, by the way, remember the data a few weeks ago? Turns out their labels are wrong, so take them out.

The Manager

The ML Engineer
And yet, even worse...

Oh, by the way, remember the data a few weeks ago? Turns out their labels are wrong, so take them out.

What???

Nooooooonnnnooo!!
How to update the model after adding, correcting, or removing data, without having to retrain the entire model?

- Add data
  - Incremental learning

- Remove data
  - Decremental learning
  - Why? – e.g., erroneous data, per user request (GDPR regulation)

- Correction to data
  - Both incremental and decremental learning
State of the Art

- Incremental learning has been considered in various contexts
  - Heuristics for deep neural networks to avoid catastrophic forgetting
  - Exact update methods for shallow models such as support vector machine (SVM)

- A few existing approaches on decremental learning
  - Not for neural networks
  - For SVM but requires old data

- We propose an exact (provably optimal) method for joint incremental and decremental learning for least squares SVM (LS-SVM) that does not require old data
## State of the Art (Details)

<table>
<thead>
<tr>
<th>Model</th>
<th>Exact?</th>
<th>Requires old data?</th>
<th>Decremental learning?</th>
</tr>
</thead>
<tbody>
<tr>
<td>[1], [2], [3]</td>
<td>SVM</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>[4]</td>
<td>SVM</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>[5]</td>
<td>SVM</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>[6], [7]</td>
<td>SVM</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>[8], [9]</td>
<td>LS-SVM</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>[10], [11]</td>
<td>LS-SVM</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Our work</td>
<td>LS-SVM</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>


Support Vector Machine (SVM)

- Binary classifier
- Can be converted into multi-class classifier using one-versus-all or one-versus-one ensemble approaches

\[ \mathbf{w}^T \mathbf{x} = 0 \]

- If \( \mathbf{w}^T \mathbf{x} > 0 \), classify as label +1
- If \( \mathbf{w}^T \mathbf{x} < 0 \), classify as label −1
Least-Squares Support Vector Machine (LS-SVM)

- Optimal solution for model parameter

\[
\mathbf{w}^* = \arg\min_{\mathbf{w} \in \mathbb{R}^j} \rho \| \mathbf{w} \|^2 + \sum_{n=1}^{N} (\mathbf{w}^T \tilde{\phi}(\mathbf{x}_n) - y_n)^2
\]

Learning loss function

\[
= \Phi [\Phi^T \Phi + \rho I_N]^{-1} \mathbf{y} = [\rho I_J + \Phi \Phi^T]^{-1} \Phi \mathbf{y}
\]  

(Aalytical optimal solution)
Main Result for Incremental/Decremental Learning

**Theorem 1.** For a given model
\[ w = \Phi [\Phi^T \Phi + \rho I_N]^{-1} y \] (3)
and auxiliary matrix
\[ C = \Phi [\Phi^T \Phi + \rho I_N]^{-1} \Phi^T, \] (4)
when adding new training data \((X_a, y_a)\) and removing existing training data \((X_r, y_r)\), we can compute the new values of \(w\) and \(C\) using
\[
\begin{aligned}
    w_{\text{new}} &= w + (C - I_J) \Phi_c \left( \rho I - \Phi_c'^T (C - I_J) \Phi_c \right)^{-1} \Phi_c'^T w - y_c \\
    C_{\text{new}} &= C + (C - I_J) \Phi_c \left( \rho I - \Phi_c'^T (C - I_J) \Phi_c \right)^{-1} \Phi_c'^T (C - I_J)
\end{aligned} \] (5)

where we define \(\Phi_c = (\Phi_a, \Phi_r), \ \Phi_c' = (\Phi_a, -\Phi_r), \ \text{and} \ y_c = (y_a, -y_r).\)
Algorithm

- Initial model training
  - Compute \( w = \Phi [\Phi^T \Phi + \rho I_N]^{-1} y \) and \( C = \Phi [\Phi^T \Phi + \rho I_N]^{-1} \Phi^T \)

- Incremental/decremental learning using new/removed data
  - Compute
    \[
    w_{\text{new}} = w + (C - I_J) \Phi_c \left( \rho I - \Phi_c^T (C - I_J) \Phi_c \right)^{-1} \Phi_c^T (w - y_c) \\
    C_{\text{new}} = C + (C - I_J) \Phi_c \left( \rho I - \Phi_c^T (C - I_J) \Phi_c \right)^{-1} \Phi_c^T (C - I_J)
    \]

- Complexity: \( O(L^3 + JL^2 + J^2 L + J^3) \)

Number of added/removed data samples in a “batch”
Dimension of feature vector
Algorithm

- Initial model training
  - Compute \( w = \Phi [\Phi^T \Phi + \rho I_N]^{-1} y \) and \( C = \Phi [\Phi^T \Phi + \rho I_N]^{-1} \Phi^T \)

- Incremental/decremental learning using new/removed data
  - Compute
    \[
    w_{\text{new}} = w + (C - I_J) \Phi_c \left( \rho I - \Phi_c^T (C - I_J) \Phi_c \right)^{-1} \Phi_c^T (w - y_c)
    \]
    \[
    C_{\text{new}} = C + (C - I_J) \Phi_c \left( \rho I - \Phi_c^T (C - I_J) \Phi_c \right)^{-1} \Phi_c^T (C - I_J)
    \]

Complexity:

- \( O(L^3 + JL^2 + J^2L + J^3) \)

Multiple data sample updates can be done either in one or multiple batches
- When using one sample per batch
  - Complexity is linear in the number updated samples
  - May not be best in practice though due to efficient implementations of matrix multiplication (details in experiments)

Number of added/removed data samples in a “batch”
Dimension of feature vector
Special Cases

- One data sample in a batch, either incremental or decremental

**Corollary 1** (Incremental learning of a single data sample). We can update $\mathbf{w}$ and $\mathbf{C}$ to include the influence of a new training data sample $(x_{N+1}, y_{N+1})$ using

$$w_{\text{new}} = w + \frac{(C - I_J)\bar{\phi}(x_{N+1})(\bar{\phi}(x_{N+1})^T w - y_{N+1})}{\bar{\phi}(x_{N+1})^T \bar{\phi}(x_{N+1}) + \rho - \bar{\phi}(x_{N+1})^T C \bar{\phi}(x_{N+1})}$$

$$C_{\text{new}} = C + \frac{(C - I_J)\bar{\phi}(x_{N+1})\bar{\phi}(x_{N+1})^T (C - I_J)}{\bar{\phi}(x_{N+1})^T \bar{\phi}(x_{N+1}) + \rho - \bar{\phi}(x_{N+1})^T C \bar{\phi}(x_{N+1})}.$$

**Corollary 2** (Decremental learning of a single data sample). We can update $\mathbf{w}$ and $\mathbf{C}$ to remove the influence of an existing training data sample $(x_r, y_r)$ using

$$w_{\text{new}} = w - \frac{(C - I_J)\bar{\phi}(x_r)(\bar{\phi}(x_r)^T w - y_r)}{-\bar{\phi}(x_r)^T \bar{\phi}(x_r) + \rho + \bar{\phi}(x_r)^T C \bar{\phi}(x_r)}$$

$$C_{\text{new}} = C - \frac{(C - I_J)\bar{\phi}(x_r)\bar{\phi}(x_r)^T (C - I_J)}{-\bar{\phi}(x_r)^T \bar{\phi}(x_r) + \rho + \bar{\phi}(x_r)^T C \bar{\phi}(x_r)}.$$
Special Cases

- Multiple data samples in a batch, either incremental or decremental

**Corollary 3** (Incremental learning of a batch of data samples). We can update $\mathbf{w}$ and $\mathbf{C}$ to include the influence of a batch of new training data samples $(\mathbf{X}_a, y_a)$ using

$$
\mathbf{w}_{\text{new}} = \mathbf{w} + (\mathbf{C} - \mathbf{I}_J) \Phi_a \left( \rho \mathbf{I} - \Phi_a^T (\mathbf{C} - \mathbf{I}_J) \Phi_a \right)^{-1} \Phi_a^T \mathbf{w} - y_a
$$

$$
\mathbf{C}_{\text{new}} = \mathbf{C} + (\mathbf{C} - \mathbf{I}_J) \Phi_a \left( \rho \mathbf{I} - \Phi_a^T (\mathbf{C} - \mathbf{I}_J) \Phi_a \right)^{-1} \Phi_a^T (\mathbf{C} - \mathbf{I}_J)
$$

where $\Phi_a = \Phi(\mathbf{X}_a)$.

**Corollary 4** (Decremental learning of a batch of data samples). We can update $\mathbf{w}$ and $\mathbf{C}$ to remove the influence of a batch of existing training data samples $(\mathbf{X}_r, y_r)$ using

$$
\mathbf{w}_{\text{new}} = \mathbf{w} - (\mathbf{C} - \mathbf{I}_J) \Phi_r \left( \rho \mathbf{I} + \Phi_r^T (\mathbf{C} - \mathbf{I}_J) \Phi_r \right)^{-1} \Phi_r^T \mathbf{w} - y_r
$$

$$
\mathbf{C}_{\text{new}} = \mathbf{C} - (\mathbf{C} - \mathbf{I}_J) \Phi_r \left( \rho \mathbf{I} + \Phi_r^T (\mathbf{C} - \mathbf{I}_J) \Phi_r \right)^{-1} \Phi_r^T (\mathbf{C} - \mathbf{I}_J)
$$

where $\Phi_r = \Phi(\mathbf{X}_r)$. 
Experiments

- MNIST dataset of handwritten digits
- Classify even/odd digits using LS-SVM
- 60,000 training data samples
- 10,000 test data samples
Training with Incorrectly Labeled Data

- Initially, 50% of data samples are mislabeled
- Removing 10% of mislabeled data increases accuracy by 12%
  - Retraining the entire model takes \(0.81\) seconds (on a personal laptop)
  - Updating using our proposed approach takes \(0.26\) seconds
Update Time with Different Batch Sizes

Log-scale

Optimal batch size from experimental (empirical) evaluation
Storage Saving

- We do not need to store original data
When we don’t exactly know the data samples to be removed…

(Removing one data sample)
Summary

▪ What we have done…
  – Provably optimal incremental and decremental learning for LS-SVM
  – Does not require old data
  – Only requires an auxiliary matrix and data samples that are added/removed
  – Reduces model update time and storage, compared to retraining
  – Preserves privacy of training data when sharing updatable models

▪ What remains to be done…
  – Incremental and decremental learning (particularly decremental learning) for generic models such as deep neural networks
    ▪ How to properly define decremental learning?
    ▪ Algorithms for joint incremental and decremental learning
    ▪ Anything provable?
    ▪ Any intuitive insights?
Thank you

Q & A
Backup slides
Application

- Decentralized learning
Application

- k-fold cross validation (incrementally replace one fold)
Proof. For the updated training dataset, we get $\Phi_{\text{new}} = (\Phi, \Phi_a, \Phi_r)$ and $y_{\text{new}} = (y, y_a, y_r)$, where $(Z_1 | Z_2)$ denotes removing the columns (of a matrix) or elements (of a vector) in $Z_2$ from $Z_1$. Using (2) and (3), we can compute the new model $w_{\text{new}}$ as

$$w_{\text{new}} = \Phi_{\text{new}} T \Phi_{\text{new}}^T + \rho I_{N_a+N_r-N_r}^{-1} y_{\text{new}}$$

$$= [\rho I_J + \Phi_{\text{new}} T \Phi_{\text{new}}^T]^{-1} \Phi_{\text{new}} y_{\text{new}}$$

$$= [\rho I_J + \Phi \Phi^T + \Phi_a \Phi_a^T - \Phi_r \Phi_r^T]^{-1} [\Phi y + \Phi_a y_a - \Phi_r y_r]$$

$$= [\rho I_J + \Phi \Phi^T + \Phi_c \Phi_c^T]^{-1} [\Phi y + \Phi_c y_c] \quad (7)$$

where we note that $\Phi_c = (\Phi_a, \Phi_r)$, $\Phi_c' = (\Phi_a, -\Phi_r)$ and $y_c = (y_a, -y_r)$ by definition.

According to the Woodbury matrix identity [24], we have

$$(A+UBV)^{-1} = A^{-1} - A^{-1} U (I+BV A^{-1} U)^{-1} B V A^{-1}.$$}

We define $A = \rho I_J + \Phi \Phi^T$, $B = 1$, $U = \Phi_c$, $V = \Phi_c^T$, and $\Psi = I + \Phi_c^T A^{-1} \Phi_c$. From (7), we can obtain the following updating process for $w$:

$$w_{\text{new}} = \left( A^{-1} - A^{-1} \Phi_c \Psi^{-1} \Phi_c^T A^{-1} \right) \left[ \Phi y + \Phi_c y_c \right]$$

$$= w + \left( A^{-1} \Phi_c y_c - A^{-1} \Phi_c \Psi^{-1} \Phi_c^T w \right)$$

$$- A^{-1} \Phi_c \Psi^{-1} \Phi_c^T A^{-1} \Phi_c y_c$$

$$= w + \left( A^{-1} \Phi_c \Psi^{-1} y_c - A^{-1} \Phi_c \Psi^{-1} \Phi_c^T w \right)$$

$$= \left( A^{-1} \Phi_c \Psi^{-1} \Phi_c^T w - y_c \right)$$

$$= w + \left( C - I_J \right) \Phi_c \left( \rho I + \Phi_c^T (C - I_J) \Phi_c \right)^{-1} \Phi_c^T (C - I_J) \Phi_c y_c$$

The last equality holds because

$$\rho A^{-1} + C = \rho \left[ \rho I_J + \Phi \Phi^T \right]^{-1} + \Phi \left[ \rho I_N + \Phi \Phi^T \right]^{-1} \Phi^T$$

$$= \rho \left[ \rho I_J + \Phi \Phi^T \right]^{-1} + \left[ \rho I_J + \Phi \Phi^T \right]^{-1} \Phi^T$$

$$= \left[ \rho I_J + \Phi \Phi^T \right]^{-1} \left[ \rho I_J + \Phi \Phi^T \right] = I.$$}

Similarly, we can compute $C_{\text{new}}$ as

$$C_{\text{new}} = \Phi_{\text{new}} T \Phi_{\text{new}}^T + \rho I_{N_a+N_r-N_r}^{-1} \Phi_{\text{new}}^T$$

$$= [\rho I_J + \Phi_{\text{new}} T \Phi_{\text{new}}^T]^{-1} \Phi_{\text{new}}^T$$

$$= [A + \Phi_a \Phi_a^T - \Phi_r \Phi_r^T]^{-1} \Phi \Phi^T + \Phi_a \Phi_a^T - \Phi_r \Phi_r^T$$

$$= [A + \Phi_c \Phi_c^T]^{-1} \left[ \Phi \Phi^T + \Phi_c \Phi_c^T \right]$$

$$= \left( A^{-1} - A^{-1} \Phi_c \Psi^{-1} \Phi_c^T A^{-1} \right) \left[ \Phi \Phi^T + \Phi_c \Phi_c^T \right]$$

$$= C + A^{-1} \Phi_c \Psi^{-1} \Phi_c^T - A^{-1} \Phi_c \Psi^{-1} \Phi_c^T C$$

$$- A^{-1} \Phi_c \Psi^{-1} \Phi_c^T A^{-1} \Phi_c \Phi_c^T$$

$$= C + A^{-1} \Phi_c \Psi^{-1} \Phi_c^T - A^{-1} \Phi_c \Psi^{-1} \Phi_c^T C$$

$$= C - A^{-1} \Phi_c \Psi^{-1} \Phi_c^T \left( C - I_J \right)$$

$$= C + \left( C - I_J \right) \Phi_c \left( \rho I + \Phi_c^T (C - I_J) \Phi_c \right)^{-1} \Phi_c^T (C - I_J).$$